The Primality Problem

- An integer p is **prime** if $p > 1$ and all positive numbers other than 1 and p itself cannot divide it.
- PRIMES asks if an integer N is a prime number.
- Dividing N by $2, 3, \ldots, \sqrt{N}$ is *not* efficient.
  - The length of N is only $\log N$, but $\sqrt{N} = 2^{0.5 \log N}$.
  - So it is an exponential-time algorithm.
- A polynomial-time algorithm for PRIMES was not found until 2002 by Agrawal, Kayal, and Saxena!
- Later, we will focus on efficient "probabilistic" algorithms for PRIMES (used in *Mathematica*, e.g.).

```
if $n = a^b$ for some $a, b > 1$ then
    return "composite";
end if
for $r = 2, 3, \ldots, n - 1$ do
    if $\gcd(n, r) > 1$ then
        return "composite";
    end if
    if $r$ is a prime then
        Let $q$ be the largest prime factor of $r - 1$;
        if $q \geq 4\sqrt{r} \log n$ and $n^{(r-1)/q} \neq 1 \bmod r$ then
            break; {Exit the for-loop.}
        end if
    end if
end for{$r - 1$ has a prime factor $q \geq 4\sqrt{r} \log n$.}
for $a = 1, 2, \ldots, 2\sqrt{r} \log n$ do
    if $(x - a)^n \neq (x^n - a) \bmod (x^r - 1)$ in $Z_n[\,x\,]$ then
        return "composite";
    end if
end for
return "prime"; {The only place with "prime" output.}
```

The Primality Problem (concluded)

- NP $\cap$ coNP is the class of problems that have succinct certificates and succinct disqualifications.
  - Each "yes" instance has a succinct certificate.
  - Each "no" instance has a succinct disqualification.
  - No instances have both.
- We will see that PRIMES $\in$ NP $\cap$ coNP.
  - In fact, PRIMES $\in$ P as mentioned earlier.

Primitive Roots in Finite Fields

Theorem 49 (Lucas and Lehmer (1927)) [a] *A number* $p > 1$ *is a prime if and only if there is a number* $1 < r < p$ *such that*

1. $r^{p-1} = 1 \bmod p$, *and*
2. $r^{(p-1)/q} \neq 1 \bmod p$ *for all prime divisors* q *of* $p - 1$.

- This r is called the **primitive root** or **generator**.
- We will prove the theorem later (see pp. 442ff).

[a] François Edouard Anatole Lucas (1842–1891); Derrick Henry Lehmer (1905–1991).

Derrick Lehmer (1905–1991)

Pratt's Theorem

Theorem 50 (Pratt (1975)) PRIMES $\in NP \cap coNP$.

- PRIMES is in coNP because a succinct disqualification is a proper divisor.
  - A proper divisor of a number n means n is *not* a prime.
- Now suppose p is a prime.
- p's certificate includes the r in Theorem 49 (p. 431).
- Use recursive doubling to check if $r^{p-1} = 1 \bmod p$ in time polynomial in the length of the input, $\log_2 p$.
  - $r, r^2, r^4, \ldots \bmod p$, a total of $\sim \log_2 p$ steps.

The Proof (concluded)

- We also need all *prime* divisors of $p-1$: $q_1, q_2, \ldots, q_k$.
  - Whether $r, q_1, \ldots, q_k$ are easy to find is irrelevant.
  - There may be multiple choices for r.
- Checking $r^{(p-1)/q_i} \neq 1 \bmod p$ is also easy.
- Checking $q_1, q_2, \ldots, q_k$ are all the divisors of $p-1$ is easy.
- We still need certificates for the primality of the q_i's.
- The complete certificate is recursive and tree-like:

$$C(p) = (r; q_1, C(q_1), q_2, C(q_2), \ldots, q_k, C(q_k)).$$

- We next prove that $C(p)$ is succinct.
- As a result, $C(p)$ can be checked in polynomial time.

The Succinctness of the Certificate

Lemma 51 *The length of $C(p)$ is at most quadratic at $5 \log_2^2 p$.*

- This claim holds when $p = 2$ or $p = 3$.
- In general, $p - 1$ has $k \leq \log_2 p$ prime divisors $q_1 = 2, q_2, \ldots, q_k$.
  - Reason:

$$2^k \leq \prod_{i=1}^{k} q_i \leq p - 1.$$

- Note also that, as $q_1 = 2$,

$$\prod_{i=2}^{k} q_i \leq \frac{p-1}{2}. \tag{4}$$

The Proof (continued)

- $C(p)$ requires:
  - 2 parentheses;
  - $2k < 2\log_2 p$ separators (at most $2\log_2 p$ bits);
  - r (at most $\log_2 p$ bits);
  - $q_1 = 2$ and its certificate 1 (at most 5 bits);
  - $q_2, \ldots, q_k$ (at most $2\log_2 p$ bits);[a]
  - $C(q_2), \ldots, C(q_k)$.

[a] Why?

The Proof (concluded)

- $C(p)$ is succinct because, by induction,

$$
\begin{aligned}
|C(p)| &\leq 5\log_2 p + 5 + 5\sum_{i=2}^{k} \log_2^2 q_i \\
&\leq 5\log_2 p + 5 + 5\left(\sum_{i=2}^{k} \log_2 q_i\right)^2 \\
&\leq 5\log_2 p + 5 + 5\log_2^2 \frac{p-1}{2} \quad \text{by inequality (4)} \\
&< 5\log_2 p + 5 + 5(\log_2 p - 1)^2 \\
&= 5\log_2^2 p + 10 - 5\log_2 p \leq 5\log_2^2 p
\end{aligned}
$$

for $p \geq 4$.

A Certificate for 23[a]

- Note that 7 is a primitive root modulo 23 and $23 - 1 = 22 = 2 \times 11$.
- So
$$C(23) = (7, 2, C(2), 11, C(11)).$$
- Note that 2 is a primitive root modulo 11 and $11 - 1 = 10 = 2 \times 5$.
- So
$$C(11) = (2, 2, C(2), 5, C(5)).$$

[a]Thanks to a lively discussion on April 24, 2008.

A Certificate for 23 (concluded)

- Note that 2 is a primitive root modulo 5 and $5 - 1 = 4 = 2^2$.
- So
$$C(5) = (2, 2, C(2)).$$
- In summary,
$$C(23) = (7, 2, C(2), 11, (2, 2, C(2), 5, (2, 2, C(2)))).$$

Basic Modular Arithmetics[a]

- Let $m, n \in \mathbb{Z}^+$.
- $m \mid n$ means m divides n; m is n's **divisor**.
- We call the numbers $0, 1, \ldots, n-1$ the **residue** modulo n.
- The **greatest common divisor** of m and n is denoted $\gcd(m, n)$.
- The r in Theorem 49 (p. 431) is a primitive root of p.
- We now prove the existence of primitive roots and then Theorem 49 (p. 431).

[a] Carl Friedrich Gauss.

Basic Modular Arithmetics (concluded)

- We use
$$a \equiv b \mod n$$
if $n \mid (a-b)$.
  - So $25 \equiv 38 \bmod 13$.
- We use
$$a = b \bmod n$$
if b is the remainder of a divided by n.
  - So $25 = 12 \bmod 13$.

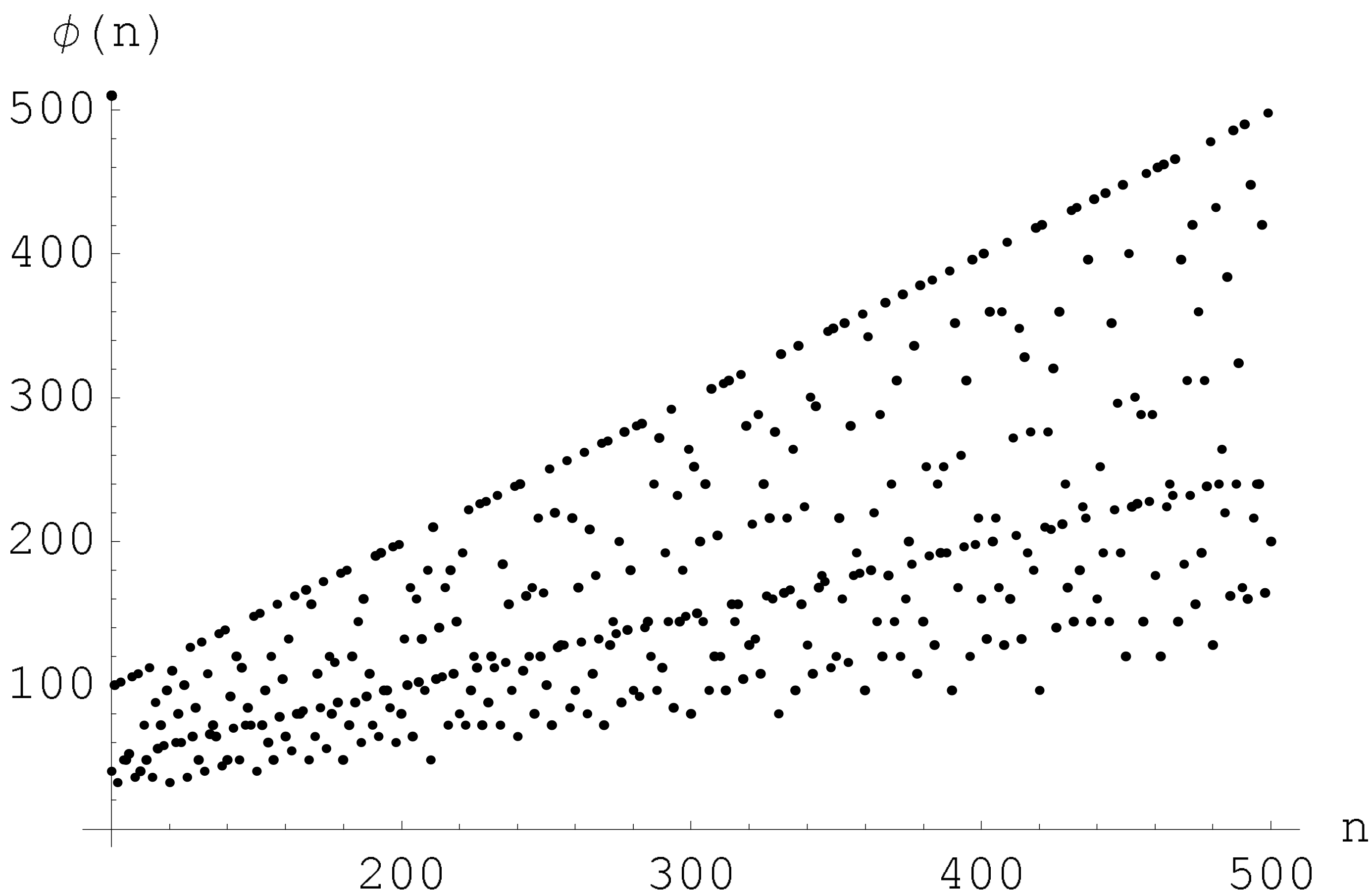
Euler's[a] Totient or Phi Function

- Let
$$\Phi(n) = \{m : 1 \leq m < n, \gcd(m, n) = 1\}$$
be the set of all positive integers less than n that are prime to n.[b]
  - $\Phi(12) = \{1, 5, 7, 11\}$.
- Define **Euler's function** of n to be $\phi(n) = |\Phi(n)|$.
- $\phi(p) = p - 1$ for prime p, and $\phi(1) = 1$ by convention.
- Euler's function is not expected to be easy to compute without knowing n's factorization.

---

[a] Leonhard Euler (1707–1783).

[b] Z_n^* is an alternative notation.



φ(n)
500
400
300
200
100
200
300
400
500
n

Two Properties of Euler's Function

The inclusion-exclusion principle[a] can be used to prove the following.

Lemma 52 $\phi(n) = n \prod_{p \mid n} (1 - \frac{1}{p})$.

- If $n = p_1^{e_1} p_2^{e_2} \cdots p_\ell^{e_\ell}$ is the prime factorization of n, then

$$\phi(n) = n \prod_{i=1}^{\ell} \left(1 - \frac{1}{p_i} \right).$$

Corollary 53 $\phi(mn) = \phi(m)\, \phi(n)$ *if* $\gcd(m, n) = 1$.

[a] Consult any textbook on discrete mathematics.

A Key Lemma

Lemma 54 $\sum_{m|n} \phi(m) = n$.

- Let $\prod_{i=1}^{\ell} p_i^{k_i}$ be the prime factorization of n and consider

$$\prod_{i=1}^{\ell} [\, \phi(1) + \phi(p_i) + \cdots + \phi(p_i^{k_i}) \,]. \tag{5}$$

- Equation (5) equals n because $\phi(p_i^k) = p_i^k - p_i^{k-1}$ by Lemma 52 (p. 444) so $\phi(1) + \phi(p_i) + \cdots + \phi(p_i^{k_i}) = p_i^{k_i}$.

- Expand Eq. (5) to yield

$$\sum_{k_1' \leq k_1, \ldots, k_\ell' \leq k_\ell} \prod_{i=1}^{\ell} \phi(p_i^{k_i'}).$$

The Proof (concluded)

- By Corollary 53 (p. 444),

$$\prod_{i=1}^{\ell} \phi(p_i^{k'_i}) = \phi\left(\prod_{i=1}^{\ell} p_i^{k'_i}\right).$$

- So Eq. (5) becomes

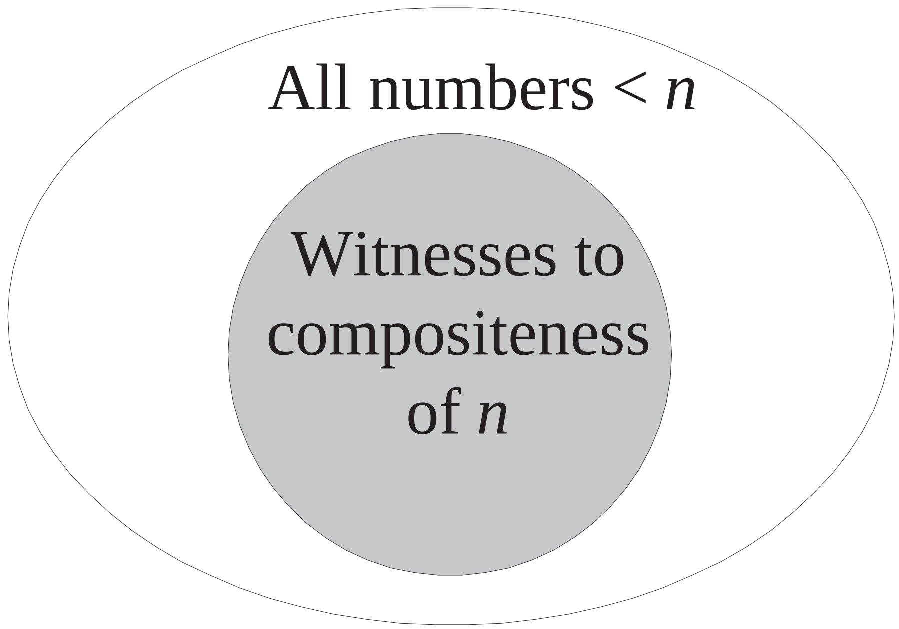
$$\sum_{k'_1 \leq k_1, \ldots, k'_\ell \leq k_\ell} \phi\left(\prod_{i=1}^{\ell} p_i^{k'_i}\right).$$

- Each $\prod_{i=1}^{\ell} p_i^{k'_i}$ is a unique divisor of $n = \prod_{i=1}^{\ell} p_i^{k_i}$.
- Equation (5) becomes

$$\sum_{m \mid n} \phi(m).$$

Leonhard Euler (1707–1783)

The Density Attack for PRIMES



The Density Attack for PRIMES

Pick $k \in \{1, \ldots, n\}$ randomly;
if $k \mid n$ and $k \neq n$ **then**
    return "n is composite";
else
    return "n is (probably) a prime";
end if

The Density Attack for PRIMES (continued)

- It works, but does it work well?
- The ratio of numbers $\leq n$ relatively prime to n (the white ring) is
$$\frac{\phi(n)}{n}.$$
- When $n = pq$, where p and q are distinct primes,
$$\frac{\phi(n)}{n} = \frac{pq - p - q + 1}{pq} > 1 - \frac{1}{q} - \frac{1}{p}.$$

The Density Attack for PRIMES (concluded)

- So the ratio of numbers $\leq n$ *not* relatively prime to n (the grey area) is $< (1/q) + (1/p)$.
  - The "density attack" has probability about $2/\sqrt{n}$ of factoring $n = pq$ when $p \sim q = O(\sqrt{n}\,)$.
  - The "density attack" to factor $n = pq$ hence takes $\Omega(\sqrt{n})$ steps on average when $p \sim q = O(\sqrt{n}\,)$.
  - This running time is exponential: $\Omega(2^{0.5 \log_2 n})$.

The Chinese Remainder Theorem

- Let $n = n_1 n_2 \cdots n_k$, where n_i are pairwise relatively prime.
- For any integers $a_1, a_2, \ldots, a_k$, the set of simultaneous equations

$$\begin{aligned} x &= a_1 \bmod n_1, \\ x &= a_2 \bmod n_2, \\ &\vdots \\ x &= a_k \bmod n_k, \end{aligned}$$

has a unique solution modulo n for the unknown x.

Fermat's "Little" Theorem[a]

Lemma 55 *For all* $0 < a < p$, $a^{p-1} = 1 \bmod p$.

- Recall $\Phi(p) = \{1, 2, \ldots, p-1\}$.
- Consider $a\Phi(p) = \{am \bmod p : m \in \Phi(p)\}$.
- $a\Phi(p) = \Phi(p)$.
  - $a\Phi(p) \subseteq \Phi(p)$ as a remainder must be between 1 and $p-1$.
  - Suppose $am = am' \bmod p$ for $m > m'$, where $m, m' \in \Phi(p)$.
  - That means $a(m - m') = 0 \bmod p$, and p divides a or $m - m'$, which is impossible.

[a] Pierre de Fermat (1601–1665).

The Proof (concluded)

- Multiply all the numbers in $\Phi(p)$ to yield $(p-1)!$.
- Multiply all the numbers in $a\Phi(p)$ to yield $a^{p-1}(p-1)!$.
- As $a\Phi(p) = \Phi(p)$, $a^{p-1}(p-1)! = (p-1)! \bmod p$.
- Finally, $a^{p-1} = 1 \bmod p$ because $p \not| (p-1)!$.

The Fermat-Euler Theorem[a]

Corollary 56 *For all $a \in \Phi(n)$, $a^{\phi(n)} = 1 \bmod n$.*

- The proof is similar to that of Lemma 55 (p. 453).
- Consider $a\Phi(n) = \{am \bmod n : m \in \Phi(n)\}$.
- $a\Phi(n) = \Phi(n)$.
  - $a\Phi(n) \subseteq \Phi(n)$ as a remainder must be between 0 and $n-1$ and relatively prime to n.
  - Suppose $am = am' \bmod n$ for $m' < m < n$, where $m, m' \in \Phi(n)$.
  - That means $a(m-m') = 0 \bmod n$, and n divides a or $m-m'$, which is impossible.

[a] Proof by Mr. Wei-Cheng Cheng (`R93922108`, `D95922011`) on November 24, 2004.

The Proof (concluded)[a]

- Multiply all the numbers in $\Phi(n)$ to yield $\prod_{m \in \Phi(n)} m$.
- Multiply all the numbers in $a\Phi(n)$ to yield $a^{\phi(n)} \prod_{m \in \Phi(n)} m$.
- As $a\Phi(n) = \Phi(n)$,

$$\prod_{m \in \Phi(n)} m = a^{\phi(n)} \left(\prod_{m \in \Phi(n)} m \right) \bmod n.$$

- Finally, $a^{\phi(n)} = 1 \bmod n$ because $n \not| \prod_{m \in \Phi(n)} m$.

[a]Some typographical errors corrected by Mr. Jung-Ying Chen (`D95723006`) on November 18, 2008.

An Example

- As $12 = 2^2 \times 3$,

$$\phi(12) = 12 \times \left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{3}\right) = 4.$$

- In fact, $\Phi(12) = \{1, 5, 7, 11\}$.
- For example,

$$5^4 = 625 = 1 \bmod 12.$$

Exponents

- The **exponent** of $m \in \Phi(p)$ is the least $k \in \mathbb{Z}^+$ such that
$$m^k = 1 \bmod p.$$
- Every residue $s \in \Phi(p)$ has an exponent.
  - $1, s, s^2, s^3, \ldots$ eventually repeats itself modulo p, say $s^i = s^j \bmod p$, which means $s^{j-i} = 1 \bmod p$.
- If the exponent of m is k and $m^\ell = 1 \bmod p$, then $k | \ell$.
  - Otherwise, $\ell = qk + a$ for $0 < a < k$, and $m^\ell = m^{qk+a} = m^a = 1 \bmod p$, a contradiction.

Lemma 57 *Any nonzero polynomial of degree k has at most k distinct roots modulo p.*

Exponents and Primitive Roots

- From Fermat's "little" theorem, all exponents divide $p-1$.
- A primitive root of p is thus a number with exponent $p-1$.
- Let $R(k)$ denote the total number of residues in $\Phi(p) = \{1, 2, \ldots, p-1\}$ that have exponent k.
- We already knew that $R(k) = 0$ for $k \not| (p-1)$.
- So
$$\sum_{k \mid (p-1)} R(k) = p - 1$$
as every number has an exponent.

Size of $R(k)$

- Any $a \in \Phi(p)$ of exponent k satisfies

$$x^k = 1 \bmod p.$$

- Hence there are at most k residues of exponent k, i.e., $R(k) \leq k$, by Lemma 57 (p. 458).
- Let s be a residue of exponent k.
- $1, s, s^2, \ldots, s^{k-1}$ are distinct modulo p.
  - Otherwise, $s^i = s^j \bmod p$ with $i < j$.
  - Then $s^{j-i} = 1 \bmod p$ with $j - i < k$, a contradiction.
- As all these k distinct numbers satisfy $x^k = 1 \bmod p$, they comprise *all* the solutions of $x^k = 1 \bmod p$.

Size of $R(k)$ (continued)

- But do all of them have exponent k (i.e., $R(k) = k$)?
- And if not (i.e., $R(k) < k$), how many of them do?
- Pick s^{ℓ}, where $\ell < k$.
- Suppose $\ell \notin \Phi(k)$ with $\gcd(\ell, k) = d > 1$.
- Then
$$(s^{\ell})^{k/d} = (s^k)^{\ell/d} = 1 \bmod p.$$
- Therefore, s^{ℓ} has exponent at most $k/d < k$.
- We conclude that
$$R(k) \leq \phi(k).$$

Size of $R(k)$ (concluded)

- Because all $p-1$ residues have an exponent,

$$p-1=\sum_{k\,|\,(p-1)} R(k) \leq \sum_{k\,|\,(p-1)} \phi(k)=p-1$$

  by Lemma 54 (p. 445).

- Hence

$$R(k)=\begin{cases}\phi(k) & \text{when } k\,|\,(p-1)\\ 0 & \text{otherwise}\end{cases}$$

- In particular, $R(p-1)=\phi(p-1)>0$, and p has at least one primitive root.
- This proves one direction of Theorem 49 (p. 431).

A Few Calculations

- Let $p = 13$.
- From p. 455, we know $\phi(p-1) = 4$.
- Hence $R(12) = 4$.
- Indeed, there are 4 primitive roots of p.
- As
$$\Phi(p-1) = \{1, 5, 7, 11\},$$
the primitive roots are
$$g^1, g^5, g^7, g^{11}$$
for any primitive root g.

The Other Direction of Theorem 49 (p. 431)

- We show p is a prime if there is a number r such that
  1. $r^{p-1} = 1 \bmod p$, and
  2. $r^{(p-1)/q} \neq 1 \bmod p$ for all prime divisors q of $p-1$.
- Suppose p is not a prime.
- We proceed to show that no primitive roots exist.
- Suppose $r^{p-1} = 1 \bmod p$ (note $\gcd(r, p) = 1$).
- We will show that the 2nd condition must be violated.

The Proof (continued)

- So we proceed to show $r^{(p-1)/q} = 1 \bmod p$ for some prime divisor q of $p-1$.
- $r^{\phi(p)} = 1 \bmod p$ by the Fermat-Euler theorem (p. 455).
- Because p is not a prime, $\phi(p) < p-1$.
- Let k be the smallest integer such that $r^k = 1 \bmod p$.
- With the 1st condition, it is easy to show that $k \,|\, (p-1)$ (similar to p. 458).
- Note that $k \,|\, \phi(p)$ (p. 458).
- As $k \leq \phi(p)$, $k < p-1$.

The Proof (concluded)

- Let q be a prime divisor of $(p-1)/k > 1$.
- Then $k|(p-1)/q$.
- By the definition of k,

$$r^{(p-1)/q} = 1 \bmod p.$$

- But this violates the 2nd condition.

Function Problems

- Decision problems are yes/no problems (SAT, TSP (D), etc.).
- **Function problems** require a solution (a satisfying truth assignment, a best TSP tour, etc.).
- Optimization problems are clearly function problems.
- What is the relation between function and decision problems?
- Which one is harder?

Function Problems Cannot Be Easier than Decision Problems

- If we know how to generate a solution, we can solve the corresponding decision problem.
  - If you can find a satisfying truth assignment efficiently, then SAT is in P.
  - If you can find the best TSP tour efficiently, then TSP (D) is in P.
- But decision problems can be as hard as the corresponding function problems.

FSAT

- FSAT is this function problem:
  - Let $\phi(x_1, x_2, \ldots, x_n)$ be a boolean expression.
  - If ϕ is satisfiable, then return a satisfying truth assignment.
  - Otherwise, return "no."
- We next show that if SAT $\in$ P, then FSAT has a polynomial-time algorithm.
- SAT is a subroutine (black box) that returns "yes" or "no" on the satisfiability of the input.

An Algorithm for FSAT Using SAT

```
t := ε; {Truth assignment.}
if φ ∈ SAT then
    for i = 1, 2, ..., n do
        if φ[ x_i = true ] ∈ SAT then
            t := t ∪ { x_i = true };
            φ := φ[ x_i = true ];
        else
            t := t ∪ { x_i = false };
            φ := φ[ x_i = false ];
        end if
    end for
    return t;
else
    return "no";
end if
```

Analysis

- If SAT can be solved in polynomial time, so can FSAT.
  - There are $\leq n+1$ calls to the algorithm for SAT.[a]
  - Boolean expressions shorter than ϕ are used in each call to the algorithm for SAT.
- Hence SAT and FSAT are equally hard (or easy).
- Note that this reduction from FSAT to SAT is not a Karp reduction (recall p. 247).
- Instead, it calls SAT multiple times as a subroutine and moves on SAT's outputs.

[a] Contributed by Ms. Eva Ou (R93922132) on November 24, 2004.

TSP and TSP (D) Revisited

- We are given n cities $1, 2, \ldots, n$ and integer distances $d_{ij} = d_{ji}$ between any two cities i and j.
- TSP (D) asks if there is a tour with a total distance at most B.
- TSP asks for a tour with the shortest total distance.
  - The shortest total distance is at most $\sum_{i,j} d_{ij}$.
    * Recall that the input string contains $d_{11}, \ldots, d_{nn}$.
    * Thus the shortest total distance is less than $2^{|x|}$ in magnitude, where x is the input (why?).
- We next show that if TSP (D) $\in$ P, then TSP has a polynomial-time algorithm.

An Algorithm for TSP Using TSP (D)

```
Perform a binary search over interval [0, 2^{|x|}] by calling
    TSP (D) to obtain the shortest distance, C;
for i, j = 1, 2, ..., n do
    Call TSP (D) with B = C and d_{ij} = C + 1;
    if "no" then
        Restore d_{ij} to old value; {Edge [i, j] is critical.}
    end if
end for
return the tour with edges whose d_{ij} ≤ C;
```

Analysis

- An edge that is not on *any* optimal tour will be eliminated, with its d_{ij} set to $C + 1$.
- An edge which is not on *all remaining* optimal tours will also be eliminated.
- So the algorithm ends with n edges which are not eliminated (why?).
- This is true even if there are multiple optimal tours![a]

[a] Thanks to a lively class discussion on November 12, 2013.

Analysis (concluded)

- There are $O(|x| + n^2)$ calls to the algorithm for TSP (D).
- Each call has an input length of $O(|x|)$.
- So if TSP (D) can be solved in polynomial time, so can TSP.
- Hence TSP (D) and TSP are equally hard (or easy).

Randomized Computation

I know that half my advertising works,
I just don't know which half.
— John Wanamaker

I know that half my advertising is
a waste of money,
I just don't know which half!
— McGraw-Hill ad.

Randomized Algorithms[a]

- Randomized algorithms flip unbiased coins.
- There are important problems for which there are no known efficient *deterministic* algorithms but for which very efficient randomized algorithms exist.
  - Extraction of square roots, for instance.
- There are problems where randomization is *necessary*.
  - Secure protocols.
- Randomized version can be more efficient.
  - Parallel algorithm for maximal independent set.[b]

---

[a] Rabin (1976); Solovay and Strassen (1977).

[b] "Maximal" (a local maximum) not "maximum" (a global maximum).

“Four Most Important Randomized Algorithms”[a]

1. Primality testing.[b]
2. Graph connectivity using random walks.[c]
3. Polynomial identity testing.[d]
4. Algorithms for approximate counting.[e]

---

[a] Trevisan (2006).
[b] Rabin (1976); Solovay and Strassen (1977).
[c] Aleliunas, Karp, Lipton, Lovász, and Rackoff (1979).
[d] Schwartz (1980); Zippel (1979).
[e] Sinclair and Jerrum (1989).